ARCOS-Speculus Detailed specification

1. Purpose and Role

The purpose of the ARCOS-Speculus is to define the Domain to use and their specific AI Agents to use. It provides the Orchestrator with the necessary information as to know how to call each Domain related Component (Speculus, Producer, Validator, Post-Processor, InputFilter)

It is one of the two parts of the ARCOS architecture that interacts with the client, the other is the Domain Speculus.

- The Orchestrator, the Producer, Validator, PostProcessor, InputFilter are not interacting with the user.
- Some clarification requests are sent to the orchestrator that are then being sent to the Domain Speculus to extract some rules from user clarification.

The ARCOS-Speculus also sets the Domain_Vocabulary, the Domain_Predefined_Rules, the Current_Project_Rules and the Domain_Schema. It load them from either a local file or a url so it can included it inline into the project file that is sent to every call to the AI Agents. The Domain-Speculus uses those files to help extract from the user's free form text the rules the process needs to deal with.

Upon modification of any selected AI Agent, a complete reset of the project occurs. If only Purpose or some other elements are changed, the process can continue as was and memory kept.

Two choices to get out of the ARCOS Speculus … Cancel or Save and Proceed. The choice of the User is sent back to the Orchestrator that then decides what's the next step.

The ARCOS-Speculus produces two files:

- ARCOS_Project.xml, containing the project ID, purpose, domain reference, and constraints (shared with all components).
- ARCOS_Domain_Agents.xml, containing the orchestrator-only agent selection and licensing information."

2. Process Flow

First select a domain for which there are available AI Agents. A domain can have many versions in their offering of each component needed for the domain.

Once the domain is selected, it is up to the user to select, for each component of the ARCOS architecture, the versions of the available AI agents (Domain Speculus, Producer, Validator, Post-Processor, FilterInput).

> Right now there are no list of available domains and their AI Agents offering so I need a Path or URL, a User, a Password, and Args to get to the parts I'll be using for the demo.

Also defined in the ARCOS-Speculus is the domain vocabulary and rules. The user indicates the location of the Domain vocabulary, the domain predefined rules, and the domain schema. They can be either local files or attained through a url … a copy of the content of the files is included inline into the project file. It also holds the current project rules that are added though the Domain-Speculus. For the BLEU example, the domain schema is the part schema for a manufacturer to upload their inventory to be sold by BLEU. The Domain is BLEU with only one offering, building a Rust CRUD interface library to be included in their manufacturing software to build the BLEU sendoff.

You can also include some Constraints for performance and security.

Once everything is defined, the user can either cancel, or save and start the Domain loop. Saving produces both ARCOS_Project.xml and ARCOS_Domain_Agents.xml. The Orchestrator uses ARCOS_Domain_Agents.xml to know which components to call, and forwards only ARCOS_Project.xml to those components.

ARCOS-Speculus serves to define the project specification before orchestration begins.

3. Communication between ARCOS-Speculus and the Orchestrator (Maestro) using schema-based XML.

How to get an agnostic orchestrator and a domain specific agents work together is to have a separation of concern and verifiable communication channels. By using XML Schema for all communications between the Orchestrator and its component, we assure that they are verifiable against a defined standard that allows the domain information to be passed around to the domain related AI Agents and keep the orchestrator agnostic.

Here are the main elements of both messaging Schemas needed for the ARCOS_Speculus agent.

Maestro_ARCOS_Speculus_Messaging.xsd

<ARCOSSpeculusMessage>		
NewProject	Choice	
<ProjectID>		
EditProject	Choice	
<ARCOSProject>	common:InlineArtifactType	
Checksum	common:SHA256Type	

ARCOS_Speculus_Response.xsd

<ARCOSSpeculusResponse>		
<Success> OR	Choice	
<ARCOSProject>	inline	Sent to all Agents by the Orchestrator
<ARCOSDomainAgents>	inline	Used only by the Orchestrator
<Timestamp>		
<Cancelled>	Choice	
<ProjectID>		
<Reason>		Optional
<Timestamp>		

The ARCOSProject element only contains Domain Specific information needed by the Domain Agents like the Domain Vocabulary, the predefined domain rules, the current project rules and the Domain Schema. It may also contain Constraints about performance and security. The ARCOSDomainAgents element contains what the Orchestrator needs to load the correct domain specific agents needed by the domain. The Orchestrator remains agnostic; it simply transfers the ARCOSProject element in all its outgoing communications with the Domain Agents.

4. Elements and Attributes of the ARCOSDomainAgents element

Element / Attribute	***Description***	***Usual Values / Notes***
<ARCOSDomainAgents>	Root of orchestrator-only agent configuration	Required
<ProjectID>	Unique project identifier	string
<DomainAgents>	Lists the agents for this project.	
<DomainSpeculus>	Defines the domain-specific speculator agent.	
<Producer>	Defines the Producer agent.	
<Validator>	Defines the Validator agent.	
<PostProcessor>	Post-Processor agent.	Optional
<FilterInput>	Pre-processor/filter agent	Optional
<Licensing>	Licensing information	Contains mode, validation, license duration

Each Domain Agents have the following elements and attributes

<%%%Agent%%%>	DomainSpeculus, Producer, Validator, PostProcessor, FilterInput
@role	DomainSpeculus, Producer, Validator, PostProcessor, FilterInput
<Execution>	
@type	app or api
<Path> or	Local executable path or,
<URL>	Remote API endpoint.
<User>	Optional credential
<Password>	Optional credential
<Args>	Optional command line arguments

The <Licensing> tag has the following elements and attributes

@mode		
<Mode>		
<Validation>		
<LicenseDuration>		

5. Elements and Attributes of the ARCOSProject element

Element / Attribute	***Description***	***Usual Values / Notes***
<ARCOSProject>	Root of project specification	Required
<ProjectID>	Unique project identifier	string
<Purpose>	Project purpose	Free text string
<DomainVocabularyAndRules>	Defines domain vocabulary, predefined rules, project rules and Domain Schema	
<DomainVocabulary>	Defines the Domain Vocabulary	Ex:BLEU_Domain_Vocabulary.xml
<DomainPredefinedRules>	List of predefined rules for the domain	Ex:BLEU_Predefined_CRUD_Rules.xml
<CurrentProjectRules>	The current project rules	
<DomainSchema>	The domain schema	Ex: BLEU_parts.xsd
<Constraints>	Project constraints	Performance, Security, flags for realtime/embedded/portable

Each DomainVocabularyAndRules elements have the following sub elements and attributes

<…Element…>		
@encoding	EncodingEnum	
<Content>	The XML inlined	
<Checksum>		

The <Constraints> tag has the following elements and attributes

@realtime	Boolean	
@embedded	Boolean	
@portable	Boolean	
<Performance>		
@level		
<Security>		
@encryption		
@policy		